Summering inför tenta och framåtblick

Human Centered Systems
Inst. för datavetenskap
Linköpings universitet

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Översikt

- Inför tentamen
- Teori- och praktikdel
- Nästa steg - Objektorienterad programmering
- Avslutande diskussion

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Kursen i sammanfattning

Dimensioner:

Grundläggande begrepp

Pythonkunskap

Hantverket & verktyg

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Läsöversikt datortenta (bild från Föreläsning 2)

- LP Part I – Part IV, kap 18
- PL: kap 1, 5.1-5.3, 5.8 intro, 6.1-6.3, 7.1-7.3, 8.1-8.3, 9.1-9.2, **11.1-11.3** (Tillägg!)
- Wikipedia kan användas som stöd för PL-avsnitten.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Regler för tentamen

- Två delar: teoridel och problemlösning
- Teoridel:
  - allmän kunskap om programmering och pythonkunskap
  - "Vad-är- och hur-förhåller-sig-x-till-y-frågor"
  - Både innehåll och formulering
- Problemlösning:
  - inlämning via SVN (troligen!)
  - Emacs som editor
  - Begränsat Unix-skal
- Krävs visst antal rätt på bägge delarna för att få godkänt

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Teoridelen

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Wikipedia-begrepp – läs sidorna (eng.)

- Källkod/Source code
- Kompilator/Compiler
- Interpretator/Interpreter
- Stegvis förfining/Stepwise refinement
- Pseudokod/pseudo code
- Flödesschema/flow chart
- KISS-principen/KISS principle
- DRY-principen/DRY principleTräd / tree (data structure)
- Bubbelsortering/bubble sort
- urvalssortering/selection sort

Läs wikipedia-sidorna översiktligt – du förväntas känna till och kunna förklara begreppen på ett bra sätt.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Begrepp ur PL-boken

Lär dig grundläggande kunskap om

- Syntax och semantik
- Tolk/kompilator/Hybrid
- Namn: Bindning, Heap, räckvidd
- Datatyper
- Uttryck: aritmetiska och Booleska, evaluering
- Tilldelningssatsen
- Kontrollflöde: kontrollpunkt
- Kontrollsatser
  - Villkor
  - Iteration
- Subprogram: funktion och procedur
- Parameteröverföring: parametrar och argument, anropsstack

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Pythonkunskap

Förståelse för syntax/semantik **för Python** inklusive:

- Variabler
- Värden/datatyper
- Filer
- Uttryck
- tilldelningssatsen
- Villkor
- Loopar
- Funktioner
- Parameteröverföring
- Högre ordningens funktioner
- Räckvidd

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Teoriuppgift 1 (generell):

Vad utmärker en Von Neumann-arkitektur? Hur förhåller sig ett Pythonprogram till denna arkitektur?

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Svar:

Von Neumann-arkitekturen består av fyra huvuddelar: I/O-enhet, CPU (processningsenhet), aritmetisk enhet och minne.
En Von Neumann-dator kör maskinkodsprogram. Python-tolken översätter Pythonprogram till maskinkod.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Teoriuppgift 2 (python):

På vilka två sätt kan man ange ett variabelt antal parametrar i en funktionsdefinition i Python? Vad innebär dessa två skrivsätt? Vilka objekttyper används i de två fallen för att representera parametrarna? Formaten är lista respektive dictionary.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Svar:

Konstruktionerna är * och **. T ex def foo(*args) och def foo(**args). * betyder att ett variabelt antal parametrar kan följa baserat på position. ** betyder att istället att de är baserade på nyckelord.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Praktiska delen

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Test av praktisk kunskap

- Test av förmåga att lösa uppgifter i Python
- Test av att du kan hantera de begrepp som introducerats i kursen
  - T ex stegvis förfining, ADTer, kunna abstrahera genom att införa funktioner
- Test att du kan lösa problem av en viss svårighetsgrad på begränsad tid
  - snabbhet = erfarenhet och kunnande
- 2007: 5 uppgifter av stigande svårighetsgrad på två timmar (i år troligen 6-7 uppgifter på ca 3 timmar)

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Programmeringsuppgift 1:

Gör en lista av frågor med svar (min 4-5) i form av tal eller enstaka ord. T ex Fråga: vad är meningen med livet; Svar: 42. Fråga: vad säger en mikrougn när den är klar; Svar: ping.

Skriv ett program som slumpar frågor till användaren. Registrera hur många frågor som ställts och hur många svar användaren har fått. Programmet svarar med att eka ut rätt svar och om de svarat rätt.

Programmet avslutas med att en viss kod ges, t ex blankt svar. Då ska programmet skriva ut hur många frågor som ställts, antal rätta svar, och hur många rätt det var i procent räknat.

Tips: använd modulen random.

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Programmeringsuppgift 2:

Gör ett program pretty_print.py som läser in en textfil och formatterar om en text efter given maximal vänster marginal. T ex om vi har följande text:

Jag läste i en gammal bok. För länge sen gick allt på tok. Jag läste: Noa, bygg en båt Snart blir det regn, då blir du våt.

ska resultatet bli följande om vi anger marginal 30:

```
Jag läste i en gammal bok.
För länge sen gick allt på
tok. Jag läste: Noa, bygg en
båt Snart blir det regn, då
blir du våt.
```

Programmet ska ta två kommandoradsargument:

```
% python pretty_print.py 40 demo.txt
```

Det första argumentet är maximal radlängd och det andra sökväg till en textfil. Programmet ska testa och ge felmeddelande om något av argumenten är av fel sort eller saknas.

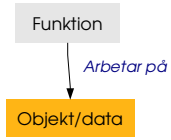
Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Näste steg: Objektorienterad programmering

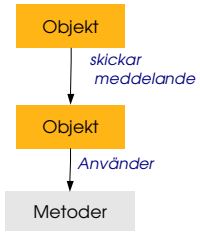
Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Objektorienterad programmering

Procedurell (objekt-baserad) programmering:



Objekt-orienterad programmering



Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Egna klasser av objekt: Person

Klassdefinition

Initieringsmetod

Metod

Metod

Metod

Metod

```
class Person:

    def __init__(self):
        self.name = None
        self.age = None

    def set_name(self, name):
        self.name = name

    def get_name(self):
        return self.name

    def set_age(self, age):
        self.age = age

    def get_age(self):
        return age
```

Jämför med ADT: samma principer gäller – men self

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Att skapa objekt och anropa dem

```
plist = []
micke = Person()
micke.set_name('Micke')
micke.set_age(42)
plist.append(micke)
bosse = Person()
bosse.set_name('Bosse')
bosse.set_age(42)
plist.append(bosse)
print "Följande namn förkommer: " + str(get_all_names(plist))
```

Skapa ett objekt av typen Person

Anropa metoder för att tilldela värden till objektet

Lägg objektet i en lista

Ett objekt till på samma sätt

```
Följande namn förkommer: ['Bosse', 'Micke']
```

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Objekten som skapas

- Objekten är muterbara, jmf en lista/dictionary/tupel
- Anrop till getName ger olika svar för de olika objekten
- Objekt har "tillstånd"
- Jämför med ADT: objekten motsvarar våra ADT-strukturer
- C++ - nästa steg i nästa period!
- Ta mer er den kunskap ni fått hittills

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License

Summering

- Teori + språkkunskap + hantverk&verktyg
- Tentamen: torsdag 18/10 kl 14-18
  - Första-person i labb-par: D316
  - Andra-person i labb-par: PC6/7
- Lycka till!!

Lars Degerstedt
Attribution-NonCommercial-ShareAlike2.5 License